Temperature upload over MQTT using Arduino UNO, ESP8266 and DHT22 sensor

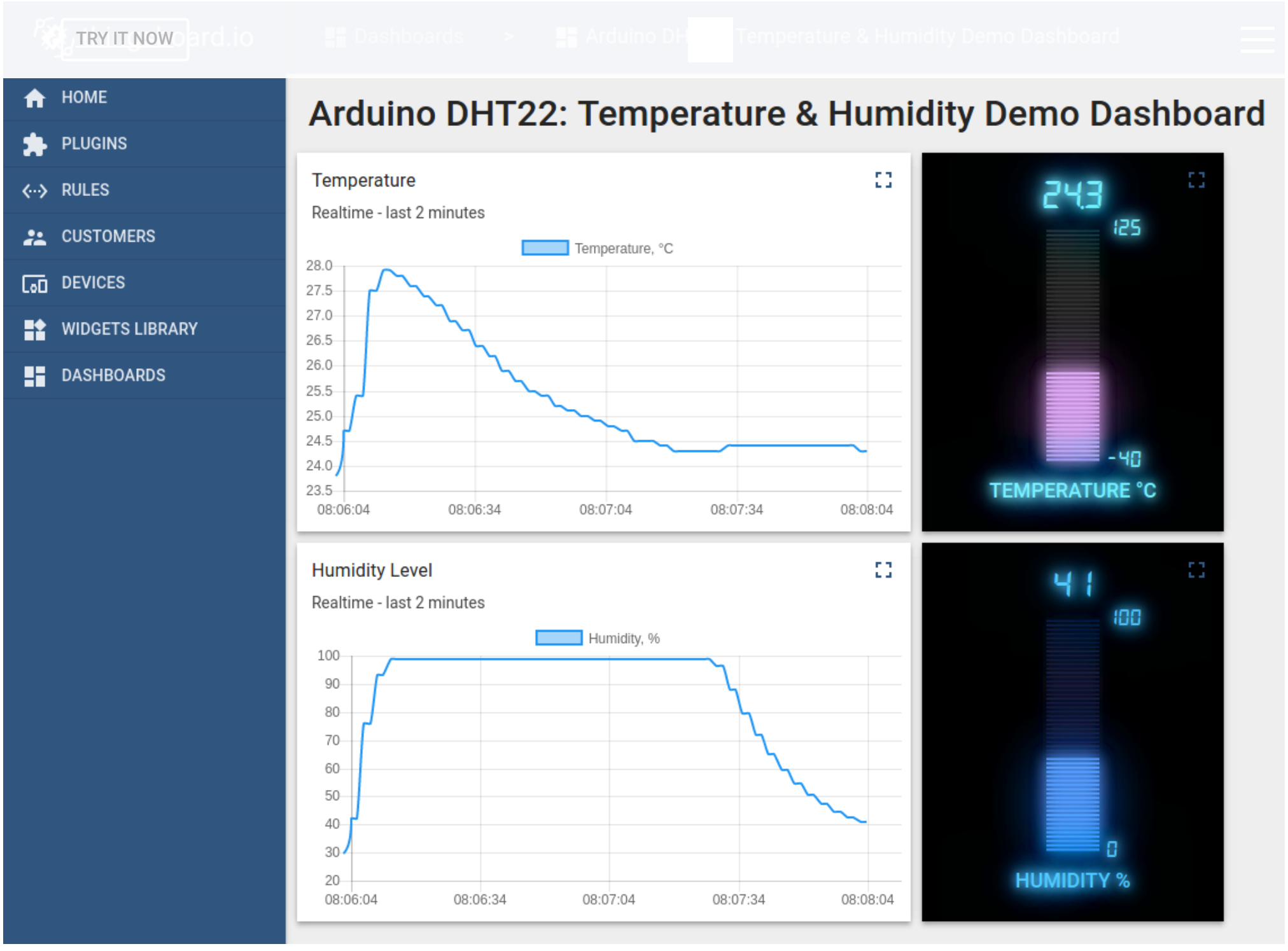
Introduction

ThingsBoard is an open-source server-side platform that allows you to monitor and control IoT devices. It is free for both personal and commercial usage and you can deploy it anywhere. If this is your first experience with the platform we recommend to review what-is-thingsboard page and getting-started guide.

This sample application performs collection of temperature and humidity values produced by DHT22 sensor and further visualization on the real-time web dashboard. Collected data is pushed via MQTT to ThingsBoard server for storage and visualization. The purpose of this application is to demonstrate ThingsBoard data collection API and visualization capabilities.

The DHT22 sensor is connected to Arduino UNO. Arduino UNO connects to the WiFi network using ESP8266. Arduino UNO pushes data to ThingsBoard server via MQTT protocol by using PubSubClient library for Arduino. Data is visualized using built-in customizable dashboard. The application that is running on Arduino UNO is written using Arduino SDK which is quite simple and easy to understand.

Once you complete this sample/tutorial, you will see your sensor data on the following dashboard.

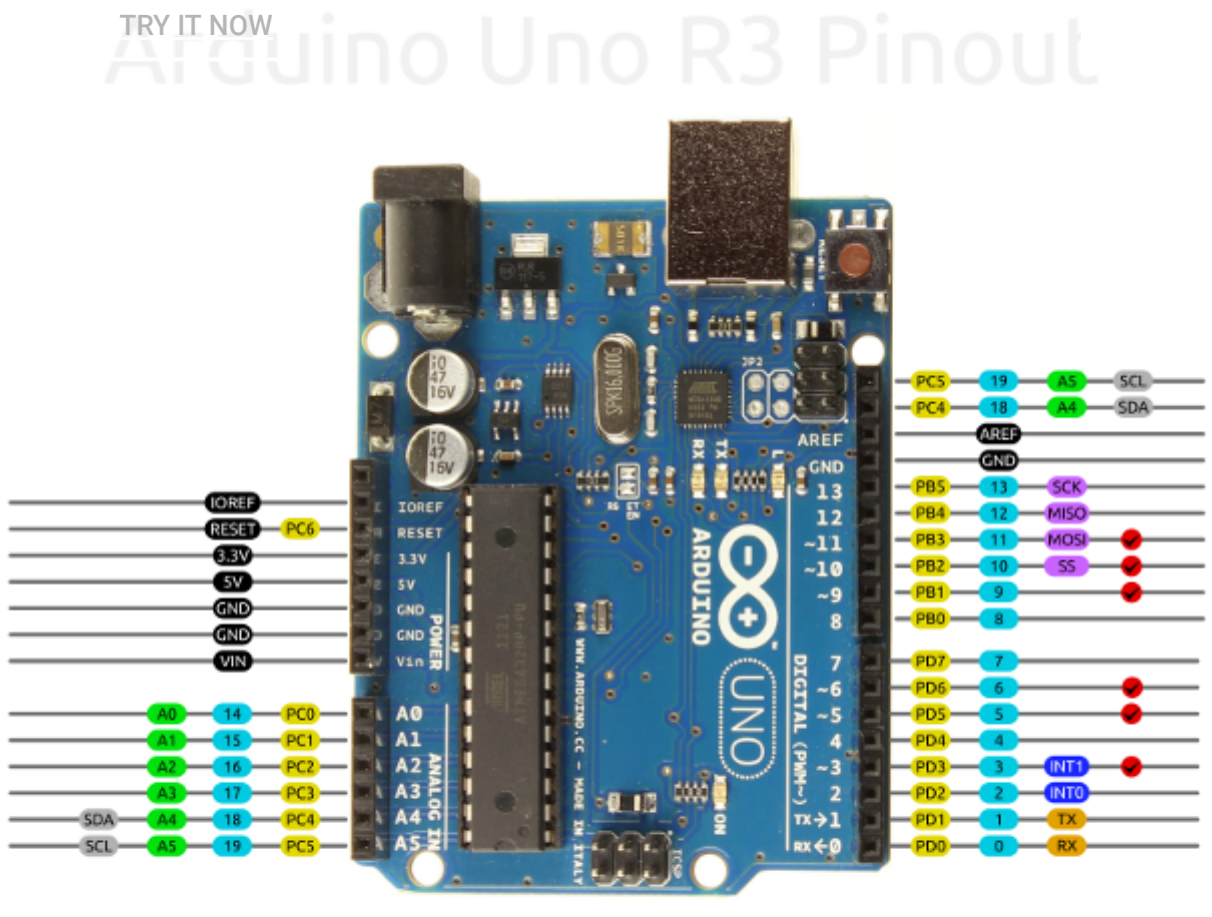


Prerequisites

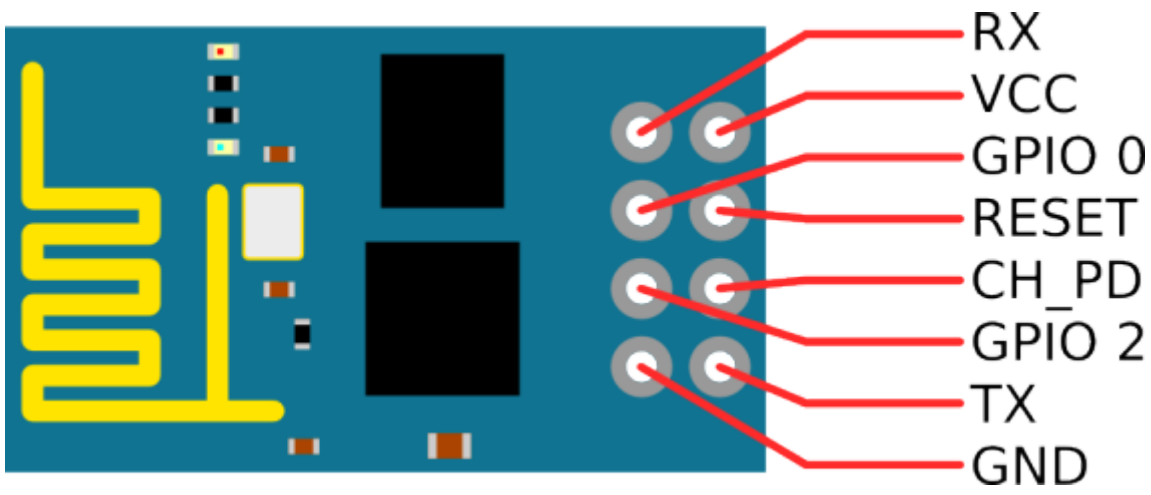
You will need to have ThingsBoard server up and running. Use either Live Demo or Installation Guide to install ThingsBoard.

List of hardware and pinouts

- Arduino UNO

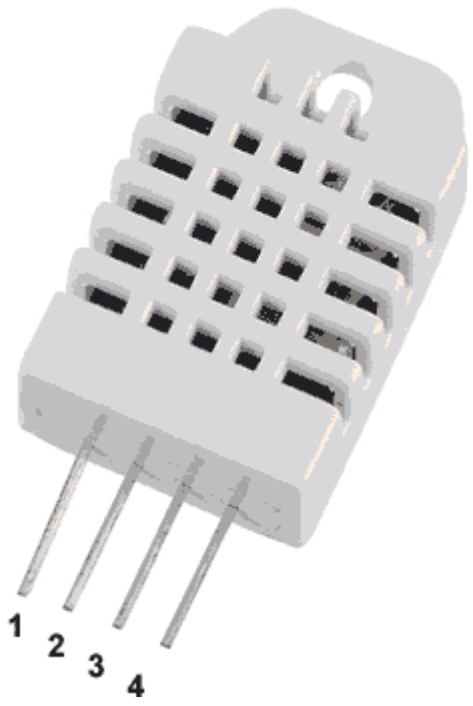


- ESP8266 module



- DHT22 sensor

DHT22 pins	
1	VCC
2	DATA
3	NC
4	GND



- Resistor (between 4.7K and 10K)
- Breadboard
- 2 female-to-female jumper wires
- 11 female-to-male jumper wires
- 3 male-to-male jumper wire

ESP8266 Firmware

TRY IT NOW

In the current tutorial WiFiEsp Arduino library is used to connect Arduino board to the internet. This library supports ESP SDK version 1.1.1 and above (AT version 0.25 and above). Please make sure that your ESP8266 has compatible firmware. You can download and flash AT25-SDK112 firmware which is tested in this tutorial.

Please note that serial baud rate of ESP8266 should be set to 9600 by the following AT command:

```
AT+UART_DEF=9600,8,1,0,0
```

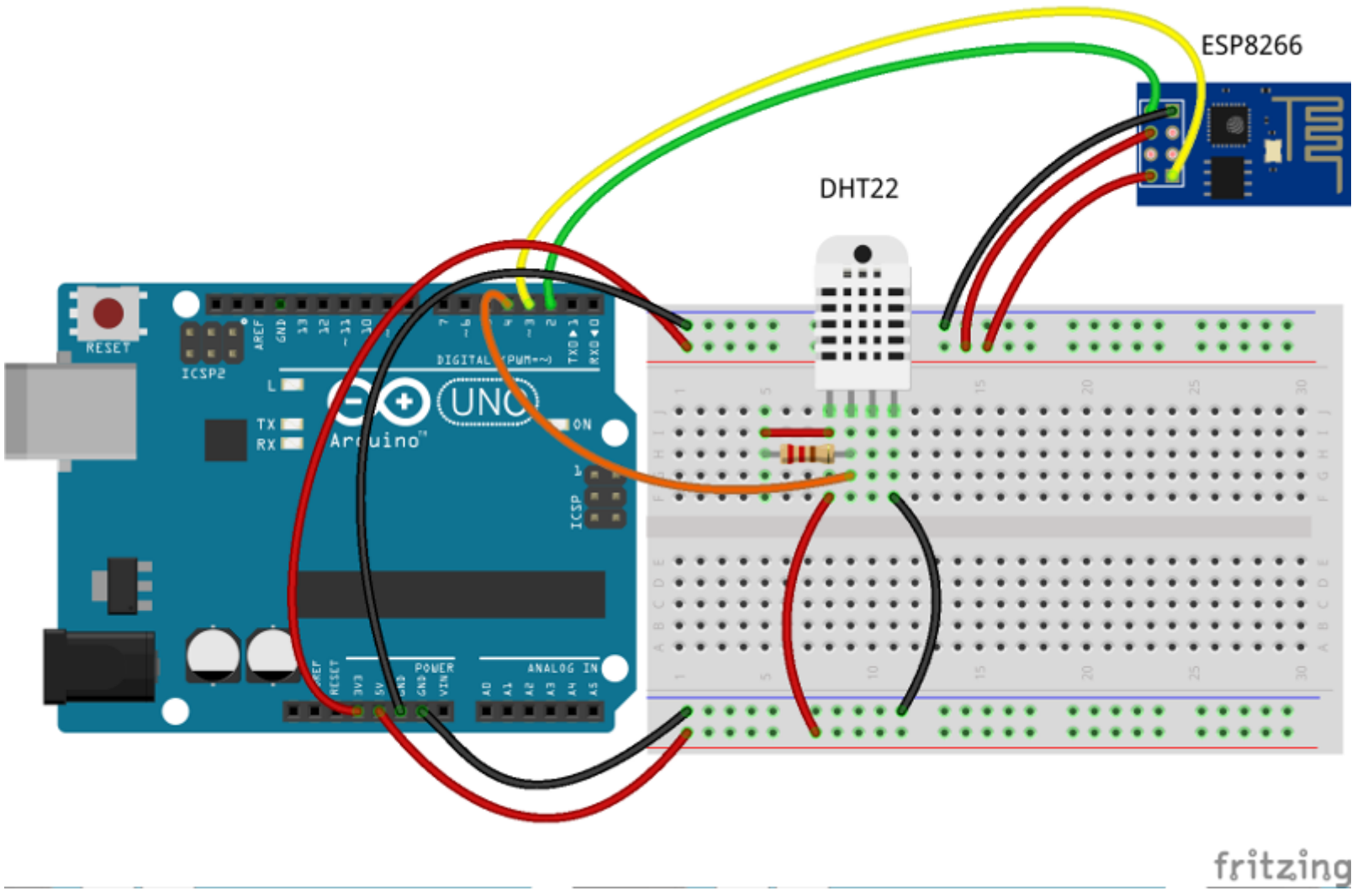
Wiring scheme

Arduino UNO Pin	ESP8266 Pin
Arduino UNO 3.3V	ESP8266 VCC
Arduino UNO 3.3V	ESP8266 CH_PD
Arduino UNO GND	ESP8266 GND (-)
Arduino UNO D2	ESP8266 RX
Arduino UNO D3	ESP8266 TX

Arduino UNO Pin	DHT-22 Pin
Arduino UNO 5V	DHT-22 VCC
Arduino UNO GND	DHT-22 GND (-)
Arduino UNO D4	DHT-22 Data

Finally, place a resistor (between 4.7K and 10K) between pin number 1 and 2 of the DHT sensor.

The following picture summarizes the connections for this project:



ThingsBoard configuration

Note ThingsBoard configuration steps are necessary only in case of **local ThingsBoard installation**. If you are using **Live Demo** instance all entities are pre-configured for your demo account. However, we recommend reviewing this steps because you will still need to get device access token to send requests to ThingsBoard.

TRY IT NOW

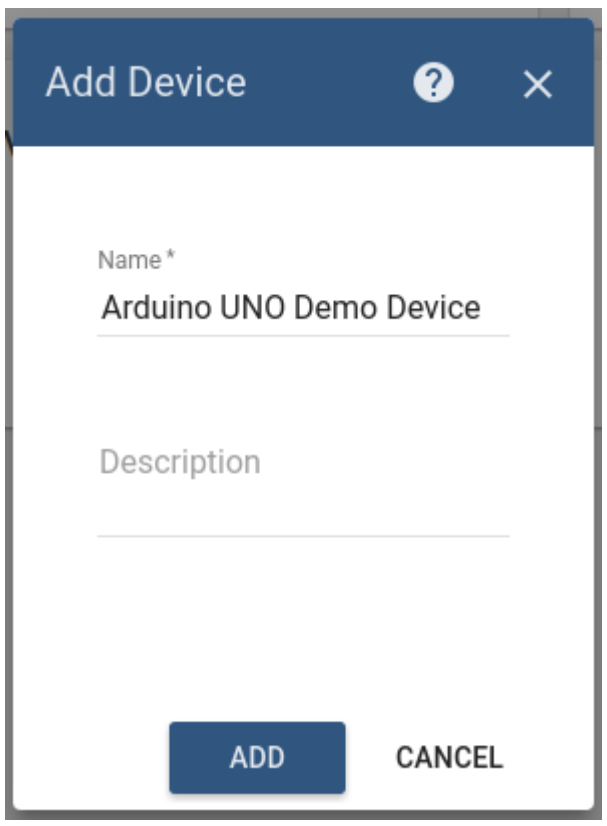
Provision your device

This step contains instructions that are necessary to connect your device to ThingsBoard.

Open ThingsBoard Web UI (http://localhost:8080) in browser and login as tenant administrator

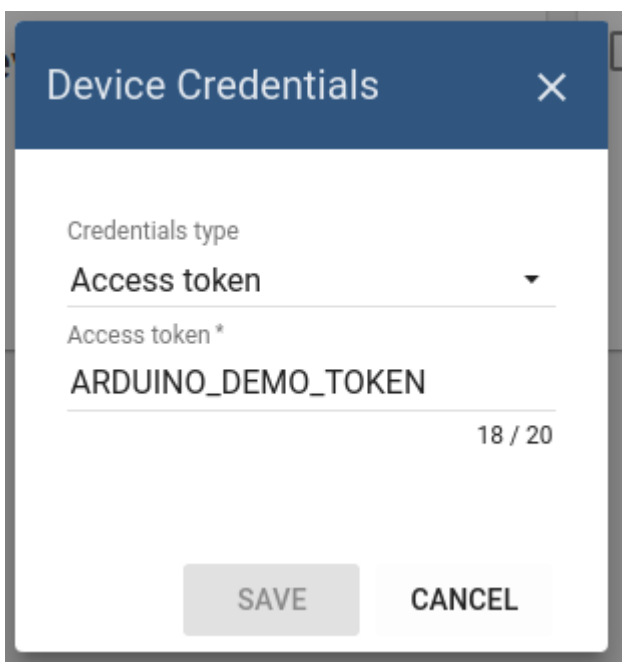
- login: tenant@thingsboard.org
- password: tenant

Go to “Devices” section. Click “+” button and create a device with the name “Arduino UNO Demo Device”.



Once device created, open its details and click “Manage credentials”.

Copy auto-generated access token from the “Access token” field. Please save this device token. It will be referred to later as **$ACCESS_TOKEN**.



Provision your dashboard

Download the dashboard file using this **link**. Use import/export **instructions** to import the dashboard to your ThingsBoard instance.

Programming the Arduino UNO device

If you already familiar with basics of Arduino UNO programming using Arduino IDE you can skip the following step and proceed with step 2.

Step 1. Arduino UNO and Arduino IDE setup.

In order to start programming the Arduino UNO device, you will need Arduino IDE and all related software installed.

Download and install Arduino IDE.

To learn how to connect your Uno board to the computer and upload your first sketch please follow this guide.

TRY IT NOW

Step 2. Install Arduino libraries.

Open Arduino IDE and go to **Sketch -> Include Library -> Manage Libraries**. Find and install the following libraries:

- PubSubClient by Nick O'Leary
- WiFiEsp by bportaluri
- Adafruit Unified Sensor by Adafruit
- DHT sensor library by Adafruit
- Arduino ThingsBoard SDK by ThingsBoard
- ArduinoJSON by bblanchon
- Arduino Http Client

Note that this tutorial was tested with the following versions of the libraries:

- PubSubClient 2.6
- WiFiEsp 2.1.2
- Adafruit Unified Sensor 1.0.2
- DHT sensor library 1.3.0
- Arduino ThingsBoard SDK 0.4
- ArduinoJSON 6.10.1
- Arduino Http Client 0.4.0

Step 3. Prepare and upload a sketch.

Download and open **arduino-dht-esp8266-mqtt.ino** sketch.

Note You need to edit following constants and variables in the sketch:

- WIFI_AP - name of your access point
- WIFI_PASSWORD - access point password
- TOKEN - the **$ACCESS_TOKEN** from ThingsBoard configuration step.
- thingsboardServer - ThingsBoard HOST/IP address that is accessible from within your wifi network. Specify "demo.thingsboard.io" if you are using live demo server.

arduino-dht-esp8266-mqtt.ino

resources/arduino-dht-esp8266-mqtt.ino

```
#include "DHT.h"
#include <WiFiEspClient.h>
#include <WiFiEsp.h>
#include <WiFiEspUdp.h>
#include "SoftwareSerial.h"
#include <ThingsBoard.h>

#define WIFI_AP "YOUR_WIFI_AP"
#define WIFI_PASSWORD "YOUR_WIFI_PASSWORD"

#define TOKEN "YOUR_ACCESS_TOKEN"

// DHT
#define DHTPIN 4
#define DHTTYPE DHT22

char thingsboardServer[] = "YOUR_THINGSBOARD_HOST_OR_IP";

// Initialize the Ethernet client object
```

TRY IT NOW

```
WiFiEspClient espClient;
// Initialize DHT sensor.
DHT dht(DHTPIN, DHTTYPE);

ThingsBoard tb(espClient);

SoftwareSerial soft(2, 3); // RX, TX

int status = WL_IDLE_STATUS;
unsigned long lastSend;

void setup() {
  // initialize serial for debugging
  Serial.begin(9600);
  dht.begin();
  InitWiFi();
  lastSend = 0;
}

void loop() {
  status = WiFi.status();
  if ( status != WL_CONNECTED) {
    while ( status != WL_CONNECTED) {
      Serial.print("Attempting to connect to WPA SSID: ");
      Serial.println(WIFI_AP);
      // Connect to WPA/WPA2 network
      status = WiFi.begin(WIFI_AP, WIFI_PASSWORD);
      delay(500);
    }
    Serial.println("Connected to AP");
  }

  if ( !tb.connected() ) {
    reconnect();
  }

  if ( millis() - lastSend > 1000 ) { // Update and send only after 1 seconds
    getAndSendTemperatureAndHumidityData();
    lastSend = millis();
  }

  tb.loop();
}

void getAndSendTemperatureAndHumidityData()
{
  Serial.println("Collecting temperature data.");

  // Reading temperature or humidity takes about 250 milliseconds!
  float humidity = dht.readHumidity();
  // Read temperature as Celsius (the default)
  float temperature = dht.readTemperature();

  // Check if any reads failed and exit early (to try again).
  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  Serial.println("Sending data to ThingsBoard:");
  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.print(" %\t");
  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.println(" *C ");

  tb.sendTelemetryFloat("temperature", temperature);
  tb.sendTelemetryFloat("humidity", humidity);
}
```

```
  TRY IT NOW
  // initialize serial for ESP module
  soft.begin(9600);
  // initialize ESP module
  WiFi.init(&soft);
  // check for the presence of the shield
  if (WiFi.status() == WL_NO_SHIELD) {
    Serial.println("WiFi shield not present");
    // don't continue
    while (true);
  }

  Serial.println("Connecting to AP ...");
  // attempt to connect to WiFi network
  while ( status != WL_CONNECTED) {
    Serial.print("Attempting to connect to WPA SSID: ");
    Serial.println(WIFI_AP);
    // Connect to WPA/WPA2 network
    status = WiFi.begin(WIFI_AP, WIFI_PASSWORD);
    delay(500);
  }
  Serial.println("Connected to AP");
}

void reconnect() {
  // Loop until we're reconnected
  while (!tb.connected()) {
    Serial.print("Connecting to ThingsBoard node ...");
    // Attempt to connect (clientId, username, password)
    if ( tb.connect(thingsboardServer, TOKEN) ) {
      Serial.println( "[DONE]" );
    } else {
      Serial.print( "[FAILED]" );
      Serial.println( " : retrying in 5 seconds" );
      // Wait 5 seconds before retrying
      delay( 5000 );
    }
  }
}
```

Connect your Arduino UNO device via USB cable and select “Arduino/Genuino Uno” port in Arduino IDE. Compile and Upload your sketch to the device using “Upload” button.

After application will be uploaded and started it will try to connect to ThingsBoard node using mqtt client and upload “temperature” and “humidity” timeseries data once per second.

Troubleshooting

When the application is running you can select “Arduino/Genuino Uno” port in Arduino IDE and open “Serial Monitor” in order to view debug information produced by serial output.

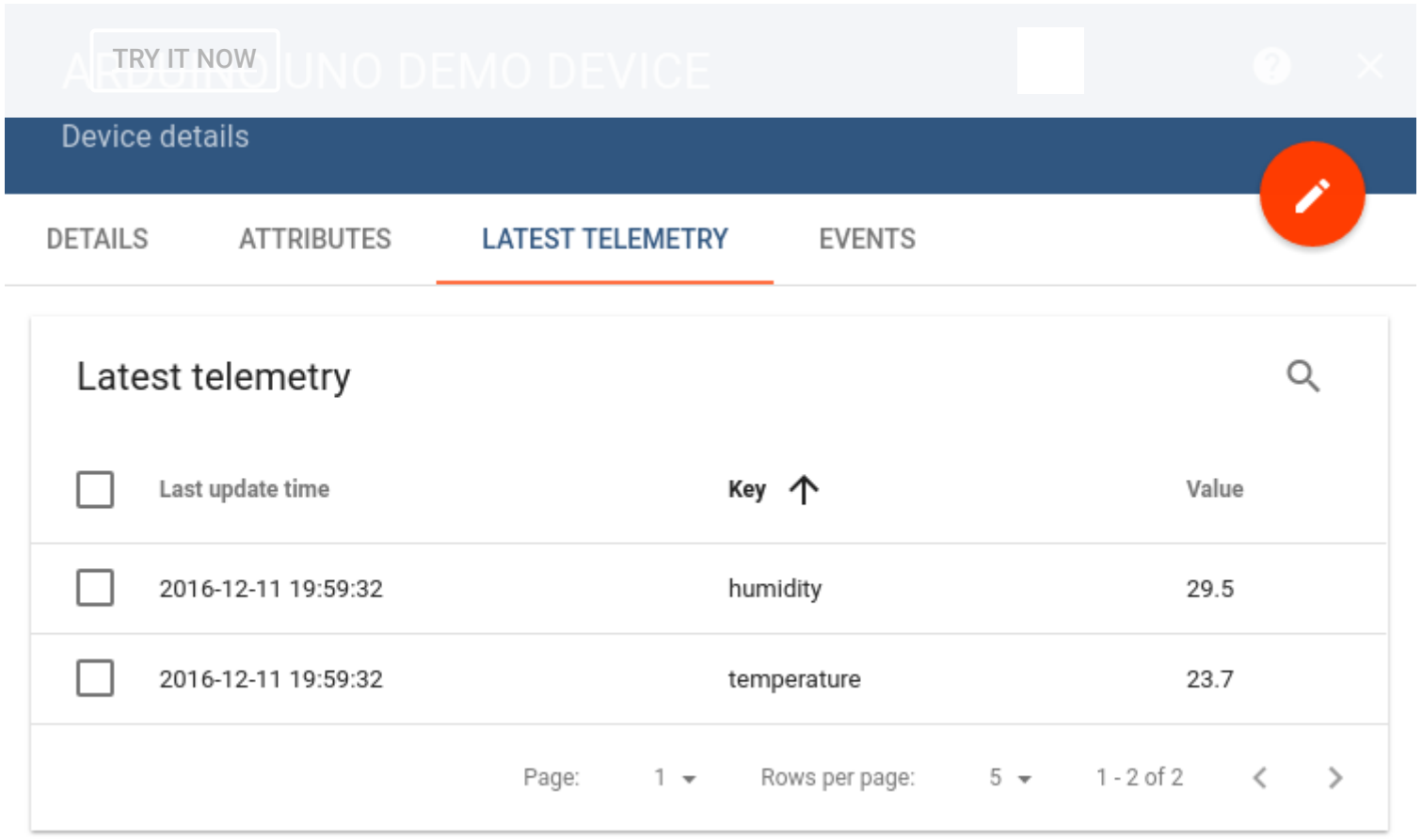
Data visualization

Finally, open ThingsBoard Web UI. You can access this dashboard by logging in as a tenant administrator. Use

- login: tenant@thingsboard.org
- password: tenant

in case of local ThingsBoard installation.

Go to **“Devices”** section and locate **“Arduino UNO Demo Device”**, open device details and switch to **“Latest telemetry”** tab. If all is configured correctly you should be able to see latest values of *“temperature”* and *“humidity”* in the table.



After, open **“Dashboards”** section then locate and open **“Arduino DHT22: Temperature & Humidity Demo Dashboard”**. As a result, you will see two time-series charts and two digital gauges displaying temperature and humidity level (similar to dashboard image in the introduction).

See also

Browse other samples or explore guides related to main ThingsBoard features:

- Device attributes - how to use device attributes.
- Telemetry data collection - how to collect telemetry data.
- Using RPC capabilities - how to send commands to/from devices.
- Rule Engine - how to use rule engine to analyze data from devices.
- Data Visualization - how to visualize collected data.

Your feedback

Don’t hesitate to star ThingsBoard on **github** to help us spread the word. If you have any questions about this sample - post it on the **issues**.

Next steps

- Getting started guides - These guides provide quick overview of main ThingsBoard features. Designed to be completed in 15-30 minutes.
- Installation guides - Learn how to setup ThingsBoard on various available operating systems.
- Connect your device - Learn how to connect devices based on your connectivity technology or solution.
- Data visualization - These guides contain instructions how to configure complex ThingsBoard dashboards.
- Data processing & actions - Learn how to use ThingsBoard Rule Engine.
- IoT Data analytics - Learn how to use rule engine to perform basic analytics tasks.
- Advanced features - Learn about advanced ThingsBoard features.
- Contribution and Development - Learn about contribution and development in ThingsBoard.

Get Started